# TITLE OF THE INVENTION

# MODULAR EXPONENTIATION CALCULATION APPARATUS AND MODULAR EXPONENTIATION CALCULATION METHOD

## CROSS-REFERENCE TO RELATED APPLICATIONS

This application is based upon and claims the benefit of priority from the prior Japanese Patent Application No. 2001-013565, filed January 22, 2001, the entire contents of which are incorporated herein by reference.

## BACKGROUND OF THE INVENTION

### 1. Field of the Invention

The present invention relates to a modular exponentiation calculation apparatus and modular exponentiation calculation method for obtaining $m = C^d \bmod (p \times q)$ with respect to object data C and independent parameters p, q, d.

### 2. Description of the Related Art

There has been proposed an algorithm and a hardware for uniting and realizing modular multiplication as a basic element for realizing algorithm (modular exponentiation calculation) of a public key cryptography with Montgomery multiplication based on a residue number system (RNS) representation which enables a parallel processing of integer operation (addition/subtraction/multiplication). This will be referred to as RNS Montgomery multiplication.

The residue number system representation (RNS

representation) will be described. For many types of public key cryptography such as an RSA cryptography, a multiple-precision integer is utilized to perform conversion, and a radix representation in which a radix is 2, so-called binary representation, is usually utilized in the representation of the multiple-precision integer. For another representation, a method of preparing a plurality of moduli $a_1$, $a_2$, …, $a_n$, and representing an integer x by a set of remainder values $x_1$, $x_2$, …, $x_n$ by these moduli as in the following equations is utilized.

$$x_1 = x \bmod a_1$$

$$x_2 = x \bmod a_2$$

…

$$x_n = x \bmod a_n$$

This representation method is called an RNS representation.

A group of moduli for use in the RNS representation will hereinafter be referred to as a base. Moreover, an element number n of the base will be referred to as a base size. The base "a" having a base size of n is represented as follows.

$$a = \{a_1, a_2, \ldots, a_n\}$$

In the RNS representation, positive integers prime to one another are usually used, and Chinese remainder theorem guarantees that the positive integer less than a product of elements of the base can uniformly be

represented by the RNS representation. That is, when the base is $a = \{a_1, a_2, \ldots, a_n\}$, and the product of elements of the base "a" is $A = a_1 \times a_2 \times \ldots \times a_n$, the positive integer less than A can be represented by the RNS representation using the base "a".

In the following, n integers x subjected to the RNS representation using the base "a" are represented by $\langle x \rangle_a$ (sometimes represented by $\langle x \rangle$ in which the base is omitted). That is, the following results.

$$\langle x \rangle_a = (x_{a1}, x_{a2}, \ldots, x_{an})$$
$$= (x \bmod a_1, x \bmod a_2, \ldots, x \bmod a_n)$$

Additionally, when two types of bases are used in the following operation, with respect to bases $a = \{a_1, a_2, \ldots, a_{n1}\}$ and $b = \{b_1, b_2, \ldots, b_{n2}\}$, aUb denotes a combination of $\{a_1, a_2, \ldots, a_{n1}\}$ and $\{b_1, b_2, \ldots, b_{n2}\}$, and $\langle x \rangle_{aUb}$ denotes the RNS representation of x by the base aUb (i.e., $\langle x \rangle_{aUb}$ denotes a combination of $\langle x \rangle_a = (x \bmod a_1, x \bmod a_2, \ldots, x \bmod a_{n1})$ and $\langle x \rangle_b = (x \bmod b_1, x \bmod b_2, \ldots, x \bmod b_{n2})$. Moreover, in the following description, for the sake of convenience two types of bases will be described as n1 = n2 = n. Additionally, n1, n2 do not have to be equal to n.

The RNS representation is advantageous in that addition, subtraction, and multiplication can easily be carried out using the product "A" of all the elements of the base. That is, desired results are obtained as results of independent addition, subtraction, and

multiplication of the respective elements by the respective moduli as follows.

$$\langle x\rangle_a + \langle y\rangle_a = (x_{a1} + y_{a1}, x_{a2} + y_{a2}, \ldots, x_{an} + y_{an})$$

$$\langle x\rangle_a - \langle y\rangle_a = (x_{a1} - y_{a1}, x_{a2} - y_{a2}, \ldots, x_{an} - y_{an})$$

$$\langle x\rangle_a \times \langle y\rangle_a = (x_{a1} \times y_{a1}, x_{a2} \times y_{a2}, \ldots, x_{an} \times y_{an})$$

Additionally, the above operations will be referred to as RNS addition, RNS subtraction, and RNS multiplication, respectively. A left side is mod A, and respective terms of a right side are mod $a_1$, mod $a_2$, …, mod $a_n$.

Therefore, n operations can be processed in parallel. When n operation units are prepared, all the operations are processed in parallel, and a fast processing is realized. Even when the number of prepared operation units is less than n, an operation speed can be enhanced in proportional to the number of units of 1 to n.

RNS Montgomery multiplication and RNS Montgomery exponentiation will next be described.

The RNS Montgomery multiplication is a method of applying a method called Montgomery multiplication to the operation in the RNS representation with respect to multiplication $\langle x\rangle_{aUb} \times \langle y\rangle_{aUb}$ with a remainder in mudulus N, and is generally carried out in the following procedure.

The RNS Montgomery multiplication is represented by $MM(\langle x\rangle_{aUb}, \langle y\rangle_{aUb}, N, aUb)$.

Here, inputs are $\langle x \rangle_{aUb}$, $\langle y \rangle_{aUb}$, N. Additionally, x and y are both less than 2N.

Bases are a, b. Additionally, x, y, N are all less than A, and less than B.

An output is $\langle w \rangle_{aUb}$. Additionally, $w = (x \times y \times B^{-1} \bmod N) + N$. Moreover, there is not +N in some case.

<Processing Content>

step-M-0: $\langle -N^{-1} \rangle_b$ is calculated.

step-M-1: $\langle s \rangle_a = \langle x \rangle_a \times \langle y \rangle_a$ is calculated.

step-M-2: $\langle s \rangle_b = \langle x \rangle_b \times \langle y \rangle_b$ is calculated.

step-M-3: $\langle t \rangle_b = \langle s \rangle_b \times \langle -N^{-1} \rangle_b$ is calculated.

step-M-4: $\langle t \rangle_b$ is base-converted to $\langle t \rangle_a$.

step-M-5: $\langle u \rangle_a = \langle t \rangle_a \times \langle N \rangle_a$ is calculated.

step-M-6: $\langle v \rangle_a = \langle s \rangle_a + \langle u \rangle_a$ is calculated.

step-M-7: $\langle w \rangle_a = \langle v \rangle_a \times \langle B^{-1} \rangle_a$ is calculated.

step-M-8: $\langle w \rangle_a$ is base-converted to $\langle w \rangle_b$.

Additionally, in the above procedure, the base conversion of the step-M-4 or step-M-8 is a processing for obtaining the RNS representation by another base (e.g., RNS representation $\langle t \rangle_a$ by a base "a") of a certain integer corresponding to the RNS representation by a certain base (e.g., integer t corresponding to RNS representation $\langle t \rangle_b$ by the base "b").

An RNS Montgomery multiplier can also realize a fast processing by increasing the operation unit for performing the processing in parallel.

Moreover, there has been proposed a method of repeatedly performing the RNS Montgomery multiplication (repeatedly utilizing the RNS Montgomery multiplier) to perform an exponentiation calculation; and constituting a cryptography processing of an RSA cryptography. This exponentiation calculation method will be referred to as the RNS Montgomery exponentiation. The RNS Montgomery exponentiation is generally carried out in the following procedure.

The RNS Montgomery exponentiation is represented by MEXP($<x>_{aUb}$, d, N, aUb).

Here, an input is $<x>_{aUb}$, exponent (binary representation) is $d = (d_k, d_{k-1}, \ldots, d_1)$, and modulus is N. Additionally $x < 2N$.

Bases are a, b. Additionally, x, N are both less than A, and less than B.

An output is $<y>_{aUb}$. Additionally, $y = x^d \times B^{-(d-1)} \bmod N$.

<Processing Content>

step-E-1: $i = k$ is set. $<y>_{aUb} = <B>_{aUb}$ is set.

step-E-2: $<y>_{aUb}$ = MM($<y>_{aUb}$, $<y>_{aUb}$, N, aUb) is calculated.

step-E-3: If $d_i = 1$, $<y>_{aUb}$ = MM ($<y>_{aUb}$, $<x>_{aUb}$, N, aUb) is calculated. If $d_i \neq 1$, nothing is carried out (nop).

step-E-4: $i = i - 1$ is set.

step-E-5: If $i = 0$, the procedure ends. If $i \neq 0$,

the procedure returns to step-E-2.

Additionally, in the above procedure, MM( ) in the step-E-2 and step-E-3 denotes the aforementioned RNS Montgomery multiplication.

A CRT modular exponentiation calculation will next be described.

For the RSA cryptography, with respect to a public key (N, e), and secret key (d, p, q), a plaintext m is enciphered into a ciphertext C with $C = m^e \bmod N$, and the ciphertext C is deciphered into the plaintext m with $m = C^d \bmod N$. Here, an exponentiation calculation method which utilizes secret prime factors p, q of a modulus N as the public key to efficiently execute decipherment, that is, which utilizes a Chinese remainder theorem (CRT) is known. This exponentiation calculation method will be referred to as the CRT modular exponentiation calculation.

<CRT Modular Exponentiation Calculation Procedure>

step-C-1: $d_p = d \bmod (p-1)$

$d_q = d \bmod (q-1)$

step-C-2: $C_p = C \bmod p$

$C_q = C \bmod q$

step-C-3: $m_p = C_p^{dp} \bmod p$

$m_q = Cq^{dq} \bmod q$

step-C-4: $m = m_p \times (q^{-1} \bmod p) \times q + m_q \times (p^{-1} \bmod q) \times p \pmod N$

Additionally, in the above procedure, since

parameters $d_p$, $d_q$, $(q^{-1} \bmod p)$, $(p^{-1} \bmod q)$ depend only on the secret key, the parameters are generally calculated beforehand and stored as a part of the secret key.

Noting that a dominant portion of a calculation amount of the CRT modular exponentiation calculation corresponds to two modular exponentiation calculations of the step-C-3, and the modular exponentiation calculation is proportional to a cube of a size of the modulus, it is seen that the calculation amount of the modular exponentiation calculation in the binary representation and CRT modular exponentiation calculation is about 1/4 (= 2/8). Additionally, when the modular exponentiation calculation of the step-C-3 is simultaneously executed in two calculation circuits, a calculation time can be expected to be reduced to about 1/8.

However, a concrete method for realizing the CRT modular exponentiation calculation of the step-C-1 to step-C-4 by the RNS Montgomery multiplication has not been realized, and it has been difficult to raise a speed of the modular exponentiation calculation of a large integer such as RSA decipherment (secret conversion).

## BRIEF SUMMARY OF THE INVENTION

According to the present invention, there is provided a modular exponentiation calculation apparatus

or modular exponentiation calculation method in which a modular exponentiation calculation is efficiently executed.

According to an embodiment of the present invention, a modular exponentiation calculation apparatus which utilizes a residue number system representation by a first base and a second base including sets of a plurality of integers with respect to object data C and parameters p, q, d (all integers included in both the bases are mutually primary, a product "A" of all the integers of the first base is $A > p$, $A > q$, a product "B" of all the integers of the second base is $B > p$, $B > q$, and $A \times B > C$) to obtain a calculation result $m = C^d \bmod (p \times q)$, the apparatus comprising:

a first processing unit configured to obtain a residue number system representation of a value $Cp^{dp} \times B \bmod p$ or a value with p added thereto based on a residue number system representation of a remainder value $Cp = C \bmod p$ by p of the data C and a remainder value $dp = d \bmod (p - 1)$ by $(p - 1)$ of the parameter d;

a second processing unit configured to obtain a residue number system representation of a value $Cq^{dq} \times B \bmod q$ or a value with q added thereto based on a residue number system representation of a remainder value $Cq = C \bmod q$ by q of the data C and a remainder value $dq = d \bmod (p - 1)$ by $(q - 1)$ of the parameter d;

a third processing unit configured to obtain a residue number system representation of an integer m' congruent with $C^d$ mod (p × q) based on both the residue number system representations obtained by the first and second processing units; and

a fourth processing unit configured to obtain the calculation result m based on a value of the integer m' obtained by converting the residue number system representation obtained by the third processing unit into a binary representation.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

FIG. 1 is a diagram showing a functional constitution example of a modular exponentiation calculation apparatus according to a first embodiment of the present invention;

FIG. 2 is a flowchart showing one example of a processing procedure of the calculation apparatus of FIG. 1;

FIG. 3 is a diagram showing an internal constitution example relating to each operation unit of the calculation apparatus of FIG. 1;

FIG. 4 is a part of the flowchart showing another example of the processing procedure of the calculation apparatus according to the embodiment in FIG. 2;

FIG. 5 is a diagram showing an internal constitution example relating to each operation unit of the modular exponentiation calculation apparatus according

to another embodiment;

FIG. 6 is a diagram showing a functional constitution example of the modular exponentiation calculation apparatus according to still another embodiment;

FIG. 7 is a diagram showing an internal constitution example relating to each operation unit of the modular exponentiation calculation apparatus according to still further embodiment; and

FIG. 8 is an explanatory view of an enciphering system using the above embodiments.

DETAILED DESCRIPTION OF THE INVENTION

An embodiment of a modular exponentiation calculation apparatus or method according to the present invention will now be described with reference to the accompanying drawings.

First Embodiment

FIG. 1 shows a functional constitution diagram of a calculation apparatus according to one embodiment of the present invention.

A calculation apparatus 1 of the present embodiment comprises an RNS operator 12 for calculating an RNS represented integer; an operator 14 for performing an auxiliary operation in a binary representation; an input/output unit 11 for performing input/output with the external device; and a controller 13 for controlling the entire constitution.

The RNS operator 12 includes an RNS inverse element calculator 122; RNS Montgomery multiplier 123; RNS Montgomery exponentiation calculator 124; RNS multiplier 125; RNS adder 126; first representation converter (binary representation to RNS representation) 127; second representation converter (RNS representation to binary representation) 128; and storage 121.

The auxiliary operator 14 in the binary representation includes a remainder calculator 141; and adder/subtracter 142.

In the aforementioned operation units, the RNS operator 12 occupies a greater part in scale.

The storage 121 is constituted, for example, of ROM and RAM for storing bases utilized in the RNS representation, parameters calculated beforehand and stored in the apparatus, and the like.

The RNS Montgomery multiplier 123 performs the aforementioned RNS Montgomery multiplication of step-M-0 to step-M-8.

The RNS Montgomery exponentiation calculator 124 performs the aforementioned Montgomery exponentiation of step-E-1 to step-E-5.

The RNS multiplier 125 performs the aforementioned RNS multiplication.

The RNS adder 126 performs the aforementioned RNS addition.

The first representation converter 127 converts a

binary representation to an RNS representation.

The second representation converter 128 converts the RNS representation to the binary representation.

Additionally, these are described in detail, for example, in Document 1 "Cox-Rower Architecture for Fast Parallel Montgomery Multiplication", Kawamura, Koike, Sano, and Shimbo, EUROCRYPT 2000、LNCS 1807, pp. 523-538, 2000.

The RNS inverse element calculator 122 calculates $\langle -x^{-1} \rangle_a$ using $\langle x \rangle_a$ as an input. That is, $-x_i^{-1}$ is calculated from $x_i$ with respect to each base $a_i$ and element $x_i$ of $\langle x \rangle_a$ (mod $a_i$). Concretely, the calculation is executed in the following procedure.

<Inverse Element Calculation in Base $a_i$>

step 0: Carmichael function $\lambda(a_i)$ is calculated with respect to the base $a_i$, and stored in the storage 121. A concrete equation of Carmichael function $\lambda$ is represented as follows. This calculation is described in "Contemporary Cryptography", Sangyo Tosyo, p. 16, authored by Tatsuaki Okamoto, Hirotsuke Yamamoto. A bit size of $\lambda(a_i)$ is not more than a bit size of $a_i$.

The following is [Fermat small theorem].

Assuming that a prime number is p, $a^{p-1} \equiv 1 \pmod{p}$ is established with respect to an arbitrary integer $a \in Z_p$ other than 0.

Based on this theorem、Euler function $\psi(n)$ with respect to an integer n is the number of elements of

$Z^*_n$. For example, when p, q have different odd numbers of elements, $\psi(p) = p-1$, $\psi(p^e) = p^{e-1}(p-1)$, $\psi(pq) = (p-1)(q-1)$.

Carmichael function $\lambda(n)$ with respect to the integer n is defined as follows. When $n = 2^{eo}p^{e1}_1, \ldots, p^{er}_r$ ($p_1, \ldots, p_r$ have different odd numbers of elements),

$$\lambda(n) = LCM(\lambda(2^{eo}), \psi(p^{e1}_1), \ldots, \psi(p^{er}_r))$$

$$\lambda(2^t) = 2^{t-1} \text{ if } t<3$$

$$= 2^{t-2} \text{ if } t \geqq 3$$

With respect to all $x(<a_i)$ prime to modulus $a_i$, $x^{\lambda(ai)} = 1 \pmod{a_i}$ is obtained. Here, the input x is assumed as secret keys p, q (prime numbers) or a product N (product of two prime numbers) of an RSA cryptography. Then, these are necessarily prime to the modulus $a_i$.

step 1: $x_i^{-1} = x_i^{\lambda(ai)-1}$ is calculated by modular multiplication in the operation unit (mod $a_i$).

step 2: $-x_i^{-1} = a_i - x_i^{-1}$ is calculated.

In the above calculation, in the step 1, the bit size of the Carmichael function $\lambda(a_i)$ is not more than the bit size of $a_i$. Therefore, when the number of words of the operation unit is set to 32 bits, the number of modular multiplication is 64 or less.

In the remainder calculator 141, a dividend x and divisor y of the binary representation are inputted, and x mod y is calculated. This calculation procedure

can be executed by usual division, and described, for example, in "The art of computer programming", Addison Wesley Longman, Inc., pp. 342-345 authored by Donald E. Knuth. The calculation amount is substantially the same as that of x1 × x2.

The adder/subtracter 142 performs binary addition/subtraction.

The calculation apparatus 1 combines the following RNS operations and executes CRT exponentiation.

·RNS Montgomery multiplication $\langle z \rangle = MM(\langle x \rangle_{a \cup b}, \langle y \rangle_{a \cup b}, p, a \cup b)$

Here, $z = x \times y \times B^{-1} \bmod p$, or

$z = (x \times y \times B^{-1} \bmod p) + p$.

·RNS Montgomery exponentiation $\langle z \rangle = MEXP(\langle x \rangle_{a \cup b}, e, p, a \cup b)$

Here, $z = x^e \times B^{-(e-1)} \bmod p$, or

$z = (x^e \times B^{-(e-1)} \bmod p) + p$.

·RNS multiplication $\langle z \rangle = MUL(\langle x \rangle_a, \langle y \rangle_a, a)$

Here, $z = x \times y \bmod A$ (multiplication of x and y in the base "a").

·RNS addition $\langle z \rangle = ADD(\langle x \rangle_a, \langle y \rangle_a, a)$

Here, $z = x + y \bmod A$ (addition of x and y in the base "a").

A last argument (a, a∪b, and the like) in the RNS operation denotes the base utilized in the RNS representation. Assuming that a value of the product of elements of the base "a" is A, and a value of the

product of elements of the base "b" is B, a value of the product of elements of the base aUb is A × B. Outputs of the RNS Montgomery multiplication and RNS Montgomery exponentiation are $z < A$ and $z < B$.

As described above, in the RNS Montgomery multiplication and RNS Montgomery exponentiation, only a value of modulus p sometimes has a large result from a property of the Montgomery multiplication. That is, MM(<x>, <y>, p, aUb) < 2p, and MEXP($<x>_{aUb}$, e, p, aUb) < 2p. When the modulus p is fixed, the output of the RNS Montgomery multiplication or the RNS Montgomery exponentiation is less than 2p, but this output can be inputted to the RNS Montgomery multiplication or the RNS Montgomery exponentiation as it is.

The following parameters are stored beforehand in the calculation apparatus 1.

Pre-registered parameters: base "a", base "b", product "A" of elements of the base "a", product "B" of elements of the base "b", product "A" × "B" of all elements of the bases "a" and "b", "$B^2$", "$<B^{-1}>_a$".

Additionally, as a relation of a parameter size in the bases "a", "b" and CRT exponentiation, at least $p < A$, $q < A$, and $p < B$, $q < B$ are necessary. As a result, with respect to $N = p \times q$, at least $N < A \times B$.

Here, the parameters inputted to the calculation apparatus 1 from the outside in order to execute the CRT exponentiation are as follows.

External input parameters: ciphertext C, $d_p$ = d mod (p-1), $d_q$ = d mod (q-1), N (= p × q), p, q, inverse element pinv = $p^{-1}$ mod q in the modulus q of p, inverse element qinv = $q^{-1}$ mod p in the modulus p of q

FIG. 2 shows one example of a processing procedure of the CRT exponentiation in the calculation apparatus 1. Moreover, FIG. 3 shows an internal constitution example relating to each operation unit of the calculation apparatus 1.

Step S0: The external input parameters C, dp, dq, N, p, q, pinv, qinv are inputted.

In the following procedure, in steps S1-p to S9-p, and S1-q to S9-q, and also in either corresponding step Si-p or Si-q, similar operation relating to two prime factors p and q of N is executed.

Step S1-p: The first representation converter 127 is utilized to convert the binary representation p to the RNS representation <p> by the base aUb ( = $<p>_a$ U $<p>_b$ = {p mod $a_1$, p mod $a_2$, …, p mod $a_n$} U {p mod $b_1$, p mod $b_2$, …, p mod $b_n$}).

Step S1-q: The first representation converter 127 is utilized to convert the binary representation q to the RNS representation <q> by the base aUb ( = $<q>_a$ U $<q>_b$ = {q mod $a_1$, q mod $a_2$, …, q mod $a_n$} U {q mod $b_1$, q mod $b_2$, …, q mod $b_n$}) by the base aUb.

Step S2-p: The RNS inverse element calculator 122 is utilized to calculate $<-p^{-1}>_b$ from $<p>_b$ obtained by

the step S1-p.

Step: S2-q: The RNS inverse element calculator 122 is utilized to calculate $<-q^{-1}>_b$ from $<q>_b$ obtained by the step S1-q.

Step S3-p: The remainder calculator 141 is utilized to calculate $bp = B^2 \bmod p$, and the first representation converter 127 is utilized to convert bp to the RNS representation <bp> by the base aUb from the binary representation.

Step S3-q: The remainder calculator 141 is utilized to calculate $bq = B^2 \bmod q$, and the first representation converter 127 is utilized to convert bq to the RNS representation <bq> by the base aUb from the binary representation.

Step S4-p: The first representation converter 127 is utilized to convert pinv to the RNS representation <pinv> by the base aUb from the binary representation.

Step S4-q: The first representation converter 127 is utilized to convert qinv to the RNS representation <qinv> by the base aUb from the binary representation.

Step S5-p: The remainder calculator 141 is utilized to calculate $Cp = C \bmod p$, and the first representation converter 127 is utilized to convert Cp to the RNS representation <Cp> by the base aUb from the binary representation.

Step S5-q: The remainder calculator 141 is utilized to calculate $Cq = C \bmod q$, and the first

representation converter 127 is utilized to convert Cq to the RNS representation <Cq> by the base aUb from the binary representation.

Step S6-p: The RNS Montgomery multiplier 123 is utilized to calculate <Cp'> = MM(<Cp>, <bp>, p, aUb). <Processing Content with Use of the aforementioned Algorithm>

step-M-1: $\langle s\rangle_a = \langle Cp\rangle_a \times \langle bp\rangle_a$ is calculated.

step-M-2: $\langle s\rangle_b = \langle Cp\rangle_b \times \langle bp\rangle_b$ is calculated.

step-M-3: $\langle t\rangle_b = \langle s\rangle_b \times \langle -p^{-1}\rangle_b$ is calculated.

step-M-4: $\langle t\rangle_b$ is base-converted to $\langle t\rangle_a$.

step-M-5: $\langle u\rangle_a = \langle t\rangle_a \times \langle p\rangle_a$ is calculated.

step-M-6: $\langle v\rangle_a = \langle s\rangle_a + \langle u\rangle_a$ is calculated.

step-M-7: $\langle Cp'\rangle_a = \langle v\rangle_a \times \langle B^{-1}\rangle_a$ is calculated.

step-M-8: $\langle Cp'\rangle_a$ is base-converted to $\langle Cp'\rangle_b$.

Thereby, RNS representation <Cp'> corresponding to either Cp' = C × B mod p or Cp' = (C × B mod p) + p is obtained.

Step S6-q: The RNS Montgomery multiplier 123 is utilized to calculate <Cq'> = MM(<Cq>, <bq>, q, aUb). Additionally, when the aforementioned algorithm is utilized, the processing content is constituted by replacing p with q in the processing content of the step S6-p.

Thereby, RNS representation <Cq'> corresponding to either Cq' = C × B mod q or Cq' = (C × B mod q) + q is obtained.

Step S7-p: The RNS Montgomery exponentiation calculator 124 is utilized to calculate <mp'> = MEXP(<Cp'>, dp, p, aUb).

<Processing Content with Use of the aforementioned Algorithm>

step-E-1: i = k is set. $\langle y \rangle_{aUb} = \langle B \rangle_{aUb}$ is set.

step-E-2: $\langle y \rangle_{aUb}$ = MM($\langle y \rangle_{aUb}$, $\langle y \rangle_{aUb}$, p, aUb) is calculated.

step-E-3: If $dp_i = 1$, $\langle y \rangle_{aUb}$ = MM($\langle y \rangle_{aUb}$, $\langle Cp' \rangle_{aUb}$, p, aUb) is calculated. If $dp_i \neq 1$, nothing is processed (nop).

Here, $dp_i$ is a value of a lower i-th bit in binary representation ($dp_k$, $dp_{k-1}$, …, $dp_1$) of dp.

step-E-4: i = i-1 is set.

step-E-5: If i = 0, the procedure ends. If i ≠ 0, the procedure returns to the step-E-2.

Thereby, RNS representation <mp'> corresponding to $mp' = Cp^{dp} \times B \bmod p$ or $mp' = (Cp^{dp} \times B \bmod p) + p$ is obtained.

Step S7-q: The RNS Montgomery exponentiation calculator 124 is utilized to calculate <mq'> = MEXP(<Cq'>, dq, q, aUb). Additionally, when the aforementioned algorithm is utilized, the processing content is constituted by replacing p with q in the processing content of the step S7-p.

Thereby, RNS representation <mq'> corresponding to either $mq' = Cq^{dq} \times B \bmod q$ or

$mq' = (Cq^{dq} \times B \bmod q) + q$ is obtained.

Step S8-p: The RNS Montgomery multiplier 123 is utilized to calculate $\langle tp \rangle = MM(\langle mp' \rangle, \langle q^{-1} \bmod p \rangle, p, aUb)$.

<Processing Content with Use of the aforementioned Algorithm>

step-M-1: $\langle s \rangle_a = \langle mp' \rangle_a \times \langle qinv \rangle_a$ is calculated.

step-M-2: $\langle s \rangle_b = \langle mp' \rangle_b \times \langle qinv \rangle_b$ is calculated.

step-M-3: $\langle t \rangle_b = \langle s \rangle_b \times \langle -p^{-1} \rangle_b$ is calculated.

step-M-4: $\langle t \rangle_b$ is base-converted to $\langle t \rangle_a$.

step-M-5: $\langle u \rangle_a = \langle t \rangle_a \times \langle p \rangle_a$ is calculated.

step-M-6: $\langle v \rangle_a = \langle s \rangle_a + \langle u \rangle_a$ is calculated.

step-M-7: $\langle tp \rangle_a = \langle v \rangle_a \times \langle B^{-1} \rangle_a$ is calculated.

step-M-8: $\langle tp \rangle_a$ is base-converted to $\langle tp \rangle_b$.

Thereby, the RNS representation $\langle tp \rangle$ corresponding to either $tp = Cp^{dp} \times q^{-1} \bmod p$ or $tp = (Cp^{dp} \times q^{-1} \bmod p) + p$ is obtained.

Step S8-q: The RNS Montgomery multiplier 123 is utilized to calculate $\langle tq \rangle = MM(\langle mq' \rangle, \langle p^{-1} \bmod q \rangle, q, aUb)$. Additionally, when the aforementioned algorithm is utilized, the processing content is constituted by replacing p with q in the processing content of the step S8-p.

Thereby, the RNS representation $\langle tq \rangle$ corresponding to either $tq = Cq^{dq} \times p^{-1} \bmod q$ or $tq = (Cq^{dq} \times p^{-1} \bmod q) + q$ is obtained.

Step S9-p: The RNS multiplier 125 is utilized to

calculate <up> = MUL(<tp>, <q>, a∪b).

Thereby, the RNS representation <up> corresponding to up = tp × q mod (A × B) is obtained.

Step S9-q: The RNS multiplier 125 is utilized to calculate <uq> = MUL(<tq>, <p>, a∪b).

Thereby, the RNS representation <uq> corresponding to uq = tq × p mod (A × B) is obtained.

Step S10: The RNS adder 126 is utilized to calculate <m'> = ADD(<up>, <uq>, a∪b).

Thereby, the RNS representation <m'> corresponding to m' = up + uq mod (A × B) is obtained.

Step S11: The second representation converter 128 is utilized to convert <m'> to the binary representation m' from the RNS representation (base a∪b).

Here, m' is not less than N in some case. Therefore, when m' is not less than N, the adder/subtracter 142 performs a processing for setting the value to be less than N.

Step S12: m' is copied to m (stored).

Step S13: m' = m' - N is calculated.

Step S14: It is determined whether or not m' < 0. Unless m' < 0, the procedure returns to the step S12. If m' < 0, the procedure comes out of a loop and shifts to step S15.

Step S15: m is outputted, and the procedure is ended.

Additionally, instead of the steps S12 to S15, for

example, other procedure such as steps S21 to S24 of FIG. 4 may be used.

Moreover, instead of inputting N from the outside, the adder/subtracter 142 may obtain N by $p \times q$.

In the procedure, in the steps S5-p, S6-p and steps S5-q, S6-q, $Cp' = C \times B \bmod p \ (+ p)$ and $Cq' = C \times B \bmod q \ (+ q)$ are calculated, and the processing corresponds to the aforementioned processing of the step-C-2 in the usual CRT exponentiation.

The processing of the steps S7-p and S7-q corresponds to the processing of step-C-3 in the usual CRT exponentiation.

The processing of the steps S8-p, S9-p, S8-q, S9-q, S10 corresponds to the processing of step-C-4 in the aforementioned usual CRT exponentiation. Here, the processing of the step-C-4 can be modified as follows, and this respect is utilized.

$$m = mp \times (q^{-1} \bmod p) \times q + mq \times (p^{-1} \bmod q) \times p$$
$$= \{mp \times (q^{-1} \bmod p) \bmod p\} \times q + \{mq \times (p^{-1} \bmod q) \bmod q\} \times p \ (\bmod N)$$

If there is no addition error of p and q in the RNS Montgomery multiplication, m' as a result of the step S11 has a relation of $m' < 2N$ in the CRT modular exponentiation calculation. Therefore, if the addition error is considered, $m' < 4N$ results. Therefore, it is necessary to subtract 3N at maximum from m', and a necessary correction is performed in the steps S12 to

S14. Since m' is converted to a binary number, it is easy to determine a positive/negative sign. This processing corresponds to the procedure for obtaining the remainder value in the modulus N in the processing of step-C-4 in the usual CRT exponentiation described in the product.

Each calculation step of the CRT modular exponentiation calculation can be executed using an operation function which can be executed by the RNS operator 12. Particularly the RNS Montgomery exponentiation of the steps S7-p and S7-q occupies a large part of the calculation processing, and it is important to utilize a sum group $a \cup b$ as a base in which bases a, b slightly larger than moduli p, q are used.

The calculation amount of the RNS Montgomery multiplication can be evaluated by the calculation amount of the base conversion executed in the multiplication. This processing requires the multiplication of the word size by an order of a base size n, when one base element is considered. Furthermore, this processing is executed for all base elements in the base to be converted. Therefore, the calculation amount of the RNS Montgomery multiplication is of the order of square of the base size n. Moreover, the calculation amount of the RNS Montgomery exponentiation corresponds to that of a processing for

repeating the RNS Montgomery multiplication by a bit size L_e of the exponent. Therefore, the calculation amount of the RNS Montgomery exponentiation is $O(n^2 \times L\_e)$.

Concretely, for example, an RSA cryptography of 1024 bits is assumed. In this case, each of secret key d, N and ciphertext C is of 1024 bits. Therefore, when this is executed in the Montgomery exponentiation in the RNS representation as in a conventional method, the base a' (and b') for use has the number of elements 33 (= 1024/32 (word size) + 1) at minimum. On the other hand, each of values Cp, Cq obtained by reducing secret keys dp, dq, p, q, C utilized in the CRT exponentiation as described in the embodiment by the moduli p, q is of 512 bits. Therefore, the base "a" (and "b") to be utilized has the number of elements 17 (= 512/32 (word size) + 1) at minimum. It is most efficient for the processing time to utilize the minimum base element number. On this assumption, the calculation amount of the modular exponentiation calculation by the CRT is compared with that of the modular exponentiation calculation which does not use the CRT. The calculation amount of the RNS Montgomery multiplication of a case in which the CRT is used is 1/4 of the calculation amount in a case in which the CRT is not used. The size of the exponent in the case in which the CRT is used is 1/2 of the calculation amount in the

case in which the CRT is not used. When the CRT is used, it is necessary to calculate the RNS Montgomery exponentiation twice. Therefore, as a whole, according to the CRT modular exponentiation calculation, RSA deciphering operation can be realized with a processing amount of about 1/4 as compared with the conventional RNS Montgomery exponentiation. Moreover, when the RNS Montgomery exponentiation is simultaneously executed in two circuits, the RSA deciphering operation can be realized at a processing amount of about 1/8 as compared with the conventional RNS Montgomery exponentiation.

As described above, according to the present embodiment, when the operation utilizing the Chinese remainder theorem, operation utilizing a residue number system, and Montgomery operation are united, the modular exponentiation calculation can be more efficiently executed.

Other embodiments will be described hereinafter.

In the procedure of FIG. 2, the procedure of the steps S1-p to S5-p may be performed in any order except that the step S2-p follows the step S1-p (the remainder calculator 141 and representation converter 127 are set to be processable in parallel, and a whole or a part of the processing may be performed in parallel).

Moreover, in the procedure of FIG. 2, in the steps Si-p and Si-q corresponding to the steps S1-p to S9-p

and S1-q to S9-q, similar operations relating to two prime factors p and q of N are executed. For the operation of S1-p to S9-p, S1-q to S9-q, p and q parts may be executed by turns. Alternatively, after all the p parts are executed, all q parts may be executed. In the latter case, since storing/retrieving an intermediate variable to/from a memory decreases, an efficiency may be enhanced.

Furthermore, the p and q parts may also be processed in a pipeline manner.

Additionally, when a whole or a part of the corresponding operation unit is set to be processable in parallel, the p and q parts can also be executed in parallel. The internal constitution example relating to each operation unit of the calculation apparatus 1 in a case in which the p and q parts are separately described is shown in FIG. 5.

Moreover, for example, all of the RNS Montgomery multiplier 123, RNS Montgomery exponentiation calculator 124, RNS multiplier 125, and RNS adder 126, only the RNS Montgomery multiplier 123 and RNS Montgomery exponentiation calculator 124, or only the RNS Montgomery exponentiation calculator 124 are set so that the processing of p parts and q parts can be performed in parallel.

Of course, each operation unit can perform a parallel calculation derived from the RNS operation and

raise the speed. In this case, the operation with respect to all the elements of the base can be constituted to be executed simultaneously, and the operation with respect to some elements of the base (e.g., the number of elements corresponding to a factor of an integer indicating the base size) can be constituted to be executed at the same time.

Moreover, in the aforementioned embodiment, an example in which pinv = $p^{-1}$ mod q, qinv = $q^{-1}$ mod p are inputted from the external device has been described, but these may be calculated from p, q. In this case, as shown in FIG. 6, as an auxiliary operation unit in the binary representation, in addition to the remainder calculator 141 and adder/subtracter 142, an inverse element calculator 143 may further be disposed.

In the inverse element calculator 131, integer x of the binary representation and value y of the modulus are inputted to calculate $x^{-1}$ mod y. This calculation is often executed by an algorithm called the extended Euclidean algorithm. The calculation is described, for example, in "The art of computer programming", Addison Wesley Longman, Inc., pp. 342-345 authored by Donald E. Knuth. In general, the calculation amount corresponds to a calculation amount of about ten modular multiplication operations having a size of y.

Furthermore, the example in which dp = d mod (p-1), dq = d mod (q-1) are inputted from the outside

has been described above in the constitution example, but may be calculated from p, q. The calculation can be performed by the remainder calculator 141.

An internal constitution example relating to each operation unit of the calculation apparatus 1 in which pinv, qinv, dp, dq are calculated from p, q is shown in FIG. 7.

Additionally, for the external input parameters (ciphertext C, dp = d mod (p-1), dq = d mod (q-1), N( = p × q), p, q, pinv = $p^{-1}$ mod q, qinv = $q^{-1}$ mod p), the parameters other than the ciphertext C are parameters corresponding to the secret key of RSA. It is also possible to store all or some of the parameters in the calculation apparatus 1. In this case, the ciphertext C and key identification information necessary for selecting a key parameter group in the calculation apparatus 1 may be inputted.

Moreover, the calculation shown in the steps S1-p to S4-p and steps S1-q to S4-q of FIG. 2 depends only on secret keys (p, q, pinv, qinv) of the RSA. However, the ciphertext C by the RSA differs with a session, but the RSA secret key is not changed very much (there can be a system in which the RSA secret key is unchanged).

Then, a result obtained by executing the steps S1-p to S4-q is stored. As long as the same RSA secret key is used, the steps S1-p to S4-q are skipped, and the result stored beforehand is utilized to perform the

processing of and after the step S5-p. When the RSA secret key is changed, the steps S1-p to S4-q may be executed anew.

Furthermore, when the RSA secret key is managed by the key identification information, the result may be associated with the key identification information and stored.

Additionally, when the RSA secret key is single and unchanged, only C is inputted from the outside, and the data (p, q, N, <p>, <q>, $<-p^{-1}>_b$, $<-q^{-1}>_b$, <bp>, <bq>, <pinv>, <qinv>, <bp>, <bq>) depending only on the RSA secret key may be stored beforehand in the storage.

Moreover, when there are a plurality of RSA secret keys, only the C and key identification information are inputted from the outside. The data (p, q, N, <p>, <q>, $<-p^{-1}>_b$, $<-q^{-1}>_b$, <bp>, <bq>, <pinv>, <qinv>, <bp>, <bq>) depending only on the RSA secret key is associated with the key identification information, and stored beforehand in the storage. The data corresponding to the key identification information inputted from the outside may be read from the storage and used.

Furthermore, when two types of bases are used, with respect to the bases $a = \{a_1, a_2, \ldots, a_{n1}\}$ and $b = \{b_1, b_2, \ldots, b_{n2}\}$, $n1 = n2 = n$ has been described, but it is also possible to set $n1 \neq n2$.

Additionally, the above-described embodiments can be applied to a communication system using an RSA

cryptography, such as shown in FIG. 8. It is more effective to apply the present invention to a decryption ($m = C^d \bmod N$) which needs more calculation amount than an encryption. But, the encryption ($C = m^e \bmod N$) is represented by an equation similar to that of the decryption. Of course, the present invention can also be applied to the encryption (e.g., a case in which the apparatus having the secret key performs the encryption). In this case, in the above description, the plaintext m is inputted instead of the ciphertext C, and the exponent e may be used instead of the exponent d.

Hardware and software constitutions of the calculation apparatus will next be described.

The present embodiment has been described assuming that the present calculation apparatus (deciphering apparatus or enciphering apparatus) is realized by hardware, but it is also possible to realize the apparatus as software.

When the apparatus is constituted as hardware, the apparatus is formed, for example, as a semiconductor apparatus, and is mounted as an operation board or card in calculators such as a personal computer in one mode. When the calculator uses OS, a driver for the operation device may be incorporated in the OS and used in the other mode. Moreover, it is also possible to form the apparatus as the semiconductor apparatus, and to

dispose the apparatus in apparatuses such as AV equipment and household electric appliances.

When the apparatus is realized by software, the apparatus can be implemented as program for allowing a computer to execute predetermined means (for allowing the computer to function as the predetermined means, or for allowing the computer to realize the predetermined function). Alternatively, the apparatus can also be implemented as a computer readable recording medium in which the program is recorded. Needless to say, it is also possible to utilize various fast techniques such as a multi-processor and pipeline processing.

According to the present invention, when the operation utilizing the Chinese remainder theorem, the operation utilizing the residue number system, and Montgomery operation are united, the modular exponentiation calculation can more efficiently be executed.

While the description above refers to particular embodiments of the present invention, it will be understood that many modifications may be made without departing from the spirit thereof. The accompanying claims are intended to cover such modifications as would fall within the true scope and spirit of the present invention. The presently disclosed embodiments are therefore to be considered in all respects as illustrative and not restrictive, the scope of the

invention being indicated by the appended claims, rather than the foregoing description, and all changes that come within the meaning and range of equivalency of the claims are therefore intended to be embraced therein. For example, other constitutions obtained by replacing a part of the illustrated constitution with another part, omitting a part of the illustrated constitution, adding another function or element to the illustrated constitution, or combining the constitutions are also possible. Moreover, another constitution logically equivalent to the illustrated constitution, another constitution including a part logically equivalent to the illustrated constitution, another constitution logically equivalent to a main part of the illustrated constitution, and the like are also possible. Furthermore, another constitution which achieves the same or similar object as the object of the illustrated constitution, another constitution which produces the same or similar effect as that of the illustrated constitution, and the like are also possible.

Additionally, it is possible to appropriately combine and implement various variations relating to various constituting parts described in the embodiment of the present invention.

Moreover, the mode for carrying out the present invention contains/includes various viewpoints, stages,

concepts, and categories such as an invention as an individual apparatus, invention relating to two or more associated apparatuses, invention as a whole system, invention relating to constituting parts inside the individual apparatus, and invention of a corresponding method.

Therefore, the present invention can be extracted from a content disclosed in the content described in the embodiment of the present invention without limiting the present invention to the illustrated constitution.

The present invention is not limited to the aforementioned modes, and can variously be modified and implemented in the technical scope.

Moreover, the present invention can also be implemented as a computer readable recording medium in which a program for allowing a computer to execute predetermined means, allowing the computer to function as predetermined means, or allowing the computer to realize a predetermined function is recorded.